HCBFS: Combining Structure-Based and TMS-Based Approaches in Model-Based Diagnosis

T. K. Satish Kumar
Knowledge Systems Laboratory
Stanford University
tksk@ksl.stanford.edu

Abstract

Model-based diagnosis can be formulated as the combinatorial optimization problem of finding an assignment of behavior modes to all the components in a system such that it is not only consistent with the system description and observations, but also maximizes the prior probability associated with it. Because the general case of this problem is exponential in the number of components, we try to leverage the structure of the physical system under consideration. Traditional dynamic programming techniques based on the underlying constraint network (like heuristics derived from maximum cardinality ordering) do not necessarily supplement or do better than algorithms based on using truth maintenance systems (like conflict-directed best first search).

In this paper, we compare the two approaches and examine how we can incorporate the dynamic programming paradigm into TMS-based algorithms to achieve the best of both the worlds. We describe an algorithm called hierarchical conflict-directed best first search (HCBFS) to solve a large diagnosis problem by heuristically decomposing it into smaller sub-problems. We also delve into some of the implications of HCBFS with respect to (1) precompiling the system description to a form that can amortize the cost of a diagnosis call and (2) facilitating other hybrid techniques for diagnosis.

1 Introduction

Diagnosis is an important component of autonomy for any intelligent agent. Often, an intelligent agent plans a set of actions to achieve certain goals. Because some conditions may be unforeseen, it is important for it to be able to reconfigure its plan depending upon the state in which it is. This mode identification problem is essentially a problem of diagnosis. In its simplest form, the problem of diagnosis is to find a suitable assignment of modes in which each component of a system is behaving in, given some observations made on it. It is possible to handle the case of a dynamic system by treating the transition variables as components (in one sense) [Kurien and Nayak, 2000].

Definition *(Diagnosis System)*: A *diagnosis system* is a triple $(SD, COMPS, OBS)$ such that:
1. SD is a system description expressed in one of several forms — constraint languages like propositional logic, probabilistic models like Bayesian network etc. SD specifies both component behavior information (SD_B) and component structure information (i.e. the topology of the system) (SD_T).
2. $COMPS$ is a finite set of components of the system. A component $comp_i$ ($1 \leq i \leq |COMPS|$) can behave in one of several, but finite set of modes (M_i). If these modes are not specified explicitly, then we assume two modes — failed ($AB(comp_i)$) and normal ($\neg AB(comp_i)$).
3. OBS is a set of observations expressed as variable values.
The task of diagnosis is to "identify" the modes in which individual components are behaving given the system description (SD) and the observations (OBS).

Definition *(Candidate)*: Given a set of integers $i_1 \cdots i_{|COMPS|}$ (such that for $1 \leq j \leq |COMPS|$, $1 \leq i_j \leq |M_j|$), a *candidate* $Cand(i_1 \cdots i_{|COMPS|})$ is defined as $Cand(i_1 \cdots i_{|COMPS|}) = (\bigcup_{k=1}^{|COMPS|}(comp_k = M_k(i_k)))$.
Here, $M_u(v)$ denotes the v^{th} element in the set M_u (assumed to be indexed in some way).

2 Diagnosis as Combinatorial Optimization

Consider diagnosing a system consisting of three bulbs B_1, B_2 and B_3 connected in parallel to the same voltage source V under the observations $off(B_1)$, $off(B_2)$ and $on(B_3)$. $AB(V) \wedge AB(B_3)$ is a diagnosis under the consistency-based formalization of diagnosis [de Kleer *et al.*, 1992] if we had constraints only of the form $\neg AB(B_3) \wedge \neg AB(V) \rightarrow B_3 = on$. Intuitively however, it does not seem reasonable because B_3 cannot be on without V working properly. One way to get around this is to include fault models in the system [Struss and Dressler, 1989]. These are constraints that explicitly describe the behavior of a component when it is not in its nominal mode (most expected mode of behavior of a component). Such a constraint in this example would be $AB(B_3) \rightarrow off(B_3)$. Diagnosis can become indiscriminate without fault models. It is also easy to see that the consistency-based approach can exploit fault models (when they are specified) to produce more intuitive diagnoses

(like only B_1 and B_2 being abnormal).

The technique of using fault models however is associated with the problem of being too restrictive. It may not model the case of some strange source of power making B_3 *on* etc. The way out of this is to allow for many modes of behavior for the components of the system. Each component has a set of modes with associated models — normal modes and fault modes. Each component has the *unknown* fault mode with the empty model. The *unknown* mode tries to capture the *modeling incompleteness* assumption (obscure modes that we cannot model in the system). Also, each mode has an associated probability that is the prior probability of the component behaving in that mode. Diagnosis can now be cast as a combinatorial optimization problem of assigning modes of behavior to each component such that it is not only consistent with $SD \cup OBS$, but also maximizes the product of the prior probabilities associated with those modes [de Kleer and Williams, 1989]. Note that the combinatorial optimization formulation of diagnosis assumes independence of the behavior modes of components.

Definition *(Combinatorial Optimization Characterization)* A candidate $H = Cand(i_1 \cdots i_{|COMPS|})$ is a diagnosis if and only if $SD \cup H \cup OBS$ is satisfiable and $P(H) = (\Pi_{k=1}^{|COMPS|} P(comp_k = M_k(i_k)))$ is maximized.

There are many other characterizations of diagnoses based on the notions of abduction, Bayesian model selection, model counting [Kumar, 2002] etc. These characterizations (including combinatorial optimization) are mostly for choosing the most likely diagnosis and do not incorporate any notion of refinement [Lucas, 1997]. The combinatorial optimization formulation to return the most likely diagnosis is however justified, practical and suited for a variety of real-life applications [Kurien and Nayak, 2000]. It also benefits from the availability of computationally efficient algorithms to solve combinatorial optimization problems [Williams and Nayak, 1996].

3 Computational Methods

Definition *(Combinatorial Optimization Problem)*: A *combinatorial optimization problem* is a tuple (V, f, c) where (1) V is a set of discrete variables with finite domains. (2) An *assignment* maps each v in V to a value in v's domain. (3) f is a function that decides *feasibility* of assignments. (4) c is a function that returns the *cost* of an assignment. (5) We want to minimize $c(V)$ such that $f(V)$ holds.

In the context of diagnosis, the following correspondences hold: (1) $V = COMPS$. (2) Domains correspond to modes of behavior of components (3) An *assignment* is a candidate. (4) c is a simple cost model assuming independence in behavior modes of components $c(comp_i = M_i(v)) = log\frac{P(comp_i=M_i(v^*))}{P(comp_i=M_i(v))}$. Here, $M_i(v^*)$ is the *nominal* mode of behavior of $comp_i$; $P(comp_i = M_i(v^*)) \geq P(comp_i = M_i(v))$ for any $v \neq v^*$. $c(Cand(i_1 \cdots i_{|COMPS|})) = \sum_{k=1}^{|COMPS|} c(comp_k = M_k(i_k))$. (5) f is the *satisfiability* of $SD \cup Cand(i_1 \cdots i_{|COMPS|}) \cup OBS$.

A brute-force method of solving such a problem is to use a simple best first search (BFS) which is clearly exponential in the number of components. It can however, be potentially improved by leveraging the structure of the system. One popular method of leveraging structure using the paradigm of dynamic programming is to use heuristics derived from a maximum cardinality ordering (m-ordering) [Tarjan and Yannakakis, 1984] over the constraint network relating the variables of the system. Such techniques have been used in a variety of domains — Bayesian network reasoning, constraint satisfaction problems [Dechter, 1992] etc. A constraint network on the variables of the system is defined by having the variables represent nodes and constraints in SD represent hyper-edges. Any kind of optimization or satisfaction defined over the variables can be done in time exponential in the induced width of the graph [Dechter, 1992]. Although the induced width itself cannot be found constructively in polynomial time, heuristics derived from m-ordering perform reasonably well in practice. Throughout the rest of this paper, we will refer to all such heuristics as naive m-ordering (naive because they do not supplement the power of TMS-based algorithms).

These heuristics however, may not be directly beneficial or applicable when the number of components is somewhat lesser than the total number of variables in the system (which is usually the case). The induced width of the constraint network relating all the variables in a physical system can easily be much more than the number of components. A further disadvantage of such approaches is that often the relationships between variables are too complex and consistency checks may involve some kind of a "simulation". Since dynamic programming techniques based on these heuristics maintain and build partial assignments, they are very likely to be costly processes. Furthermore, in many cases, the number of faulty components is usually far lesser than the total number of components and these techniques do not exploit this significantly towards computational gains.

One approach that addresses these problems somewhat indirectly is conflict-directed best first search (CBFS) [Williams and Nayak, 1996]. It is based on the idea of examining hypotheses in decreasing order of their prior probabilities and using a truth maintenance system (TMS) to catch minimal conflicts and focus the search. QCBFS [Kumar, 2001] is an extension of CBFS that leverages qualitative knowledge present in the system. Because hypotheses are examined in order of their probabilities, diagnoses that entail a nominal behavior for all but a few components are caught as soon as possible (unlike in the naive m-ordering case).

A TMS incorporates and uses the following properties: (1) If a partial assignment to the mode behaviors of a subset of the components is inconsistent, then any other assignment that contains this subset unchanged is also inconsistent. (2) Smaller conflicts result in more pruning of the search space and therefore, whenever an assignment A is infeasible, a minimal infeasible subset of A is returned (using dependency tracking). (3) Since the hypotheses that we examine differ only incrementally from one another in the assignments for behavior modes of components, feasibility checks are made more efficient (like in ITMS [Nayak and Williams, 1997]).

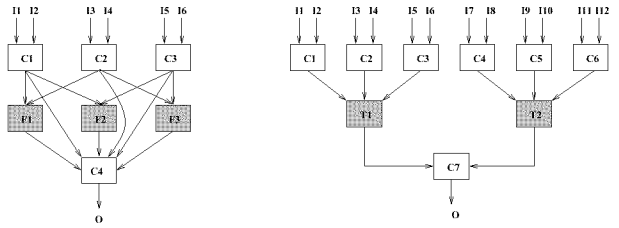

Figure 1: (a) Shows the worst-case scenario for m-ordering. (b) Shows the worst-case scenario for CBFS.

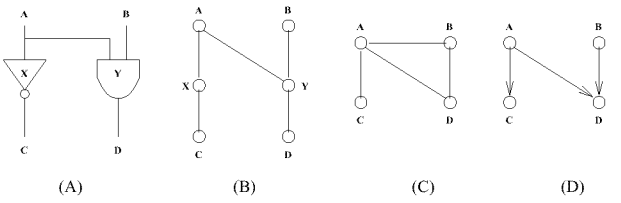

Figure 2: (A) The physical setting. (B) The graph representation. (C) The constraint network. (D) The *T-Graph.*

3.1 Comparison of naive m-ordering and CBFS

While naive m-ordering exploits the structure of the underlying constraint network, it does not exploit the fact that we are interested in an assignment only to the components of the system (and not the intermediate variables). This becomes a liability especially when consistency checks involve "simulation" and are therefore costly. It performs badly when a "small" number of components are "tightly" connected. Figure 1(a) illustrates the bad behavior of m-ordering. There are 4 components that can possibly behave in different modes (C1, C2, C3 and C4). F1, F2 and F3 are not modeled as components but are some complex mappings (involving simulation) from their inputs to outputs. The number of parents of C4 is equal to 6 and the combinatorial optimization problem is exponential in this quantity [Darwiche, 1998]. A TMS-based algorithm however, would require only a search space exponential in the number of components (=4). This can be verified by noting that once a set of modes is assumed for each component (as in a TMS-based algorithm), verifying that the current set of inputs lead to the observations is not exponential but only polynomial in the size of the graph. This is because any component maps its inputs to a unique output and we just need to follow the inputs through all the transformations defined by the components to eventually verify whether there is a match with the observations. In the case of naive m-ordering however, combinatorial optimization requires us to compute and store against all values of communication variables around a family (also called partition), the most likely modes of behavior of the components in it. This makes it exponential in the induced width of the graph. It is also easy to see (as claimed earlier) that when the diagnosis is quite close to the nominal behaviors of components, there is no obvious way of exploiting it with m-ordering.

CBFS on the other hand, exploits the fact that we are interested in an assignment only for the components of the system, but does not exploit the structure of the physical setting efficiently. The only indirect way in which the structure comes into play is in the TMS implementation of f to catch minimal conflicts. The problem with CBFS is in large due to the fact that all inconsistencies are traced back to the components. This makes CBFS perform sub-optimally when components are "loosely" connected. Figure 1(b) illustrates the bad behavior of CBFS. An observation of O = 1 when C7 is an XOR gate entails the conflicts $\{T1 = 1, T2 = 1\}$ and $\{T1 = 0, T2 = 0\}$. Note that T1 = 0, T2 =0, T1 = 1 or T2 = 1 are not conflicts by themselves. If all inconsistencies are traced back to the components C1 - C6 however, the search space over component behavior modes is never pruned by a minimal conflict of size lesser than 6. If on the other hand, we split the problem into two (by treating the cases $\{T1 = 1, T2 = 0\}$ and $\{T1 = 0, T2 = 1\}$ separately) the search space can be reduced to being exponential in 4 variables (rather than 6).

4 Hierarchical Conflict-Directed Best First Search (HCBFS)

Before we describe HCBFS as an algorithm that can combine the best of both the above approaches, we define the following notions related to the structure of a physical setting.

Definition *(Structural Parameter Set)*: The *structural parameter set* S of a physical system is the 4-tuple $S = (COMPS, I, O, T)$. Here, I is the set of external inputs, O is the set of output variables under observation, and T is the set of intermediate variables in the system which are not under observation.

Definition *(Graph Representation)*: The *graph representation* of a physical system with structural parameter set S and a topology characterized by SD_T is a graph with nodes corresponding to elements in S and undirected edges corresponding to physical connections inferred from SD_T.

Definition *(*-node)*: A node in the graph representation of a physical system is a *c-node*, *i-node*, *o-node* or a *t-node* when it corresponds respectively to a component, input variable, output variable or an intermediate variable.

Definition *(T-Graph)*: The *T-Graph* of a physical system with structural parameter set S and topology SD_T is a graph built out of removing the *c-nodes* from its graph representation and directly connecting the inputs to their outputs (in that direction).

Figure 2 illustrates the above definitions for a simple physical setting. Note that the *graph representation* is not the same as the constraint network specified by SD. While the constraint network is built on the variables of the system (excluding components) using SD, the graph representation is built only out of SD_T (and includes the components). The *T-Graph* represents the *causal* relationships among the variables (excluding the components) and it can be observed that the constraint network is equivalent to the *T-Graph* moralized by making a clique out of all the parents of any node [Dechter, 1992].

Notation: Let $M(i)$ be the set of modes in which component $comp_i$ can behave. Let c_i be the cardinality of this set. Let $T(i)$ be the set of values an intermediate variable *t-node*$_i$ can

take. Let t_i be the cardinality of this set.

Definition *(c-size)*: The *c-size* of a sub-graph G is the product of the number of modes in which each component it contains can behave, = $\Pi_{i \in COMPS(G)} c_i$.

Definition *(t-partition)*: A *t-partition* of a graph representation is any collection of vertex induced sub-graphs $S_1 \cdots S_k$ such that for all i, j with $1 \leq i, j \leq k$, $S_i \cap S_k \subseteq T$.

Definition *(t-size)*: The *t-size* of a sub-graph in a *t-partition* of the original graph is the product of the number of different values each of the *t-nodes* it shares with other sub-graphs, can take. In other words, suppose $S_1 \cdots S_k$ form a *t-partition* of the original graph. Denote the *t-nodes* in each of these sub-graphs by $ST_1 \cdots ST_k$. The *t-size* of S_i is given by $\Pi_{j \in ST_i}(t_j | \exists h, 1 \leq h \leq k, h \neq i, j \in ST_h)$.

Definition *(ct-size)*: The *ct-size* of a graph is the product of its *c-size* and *t-size*.

Given the *graph representation* of a physical system, its *c-size* characterizes the size of the search space for CBFS. The general idea behind HCBFS is to reduce the effective search space of CBFS using dynamic programming. Suppose we were able to divide the system into two subsystems that had components $comp_{i_1} \cdots comp_{i_{n_1}}$ and $comp_{j_1} \cdots comp_{j_{n_2}}$ such that $n_1 + n_2 = |COMPS|$. Now, the search space for each of these two individual partitions (for CBFS) becomes their respective *c-sizes*. Calling them C_1 and C_2 respectively, we have $C_1.C_2 = C$ (C is the *c-size* of the original graph). Of course, the search cannot simply be done in each of them independently because of the common variables they share. However, we can apply the idea of dynamic programming to solve each of these partitions for *all* values of the variables they share and then "join" the two results. If we allow for the common variables to be only among the *t-nodes*, then the size of the search space becomes $C_1T + C_2T + T^2$ (T is the *t-size* of the common *t-nodes*). $C_1T + C_2T$ accounts for solving the sub-problems for all values of the communication variables, and T^2 accounts for "joining" them. It should be noted however, that if consistency checks involve "simulation", then the T^2 term tends to be negligible (because search over the join-space does not involve simulation). Generalizing the above idea of dynamic programming, it is also possible to characterize n-way splits which partition the original graph into n partitions each of which share communication variables with a subset of the others.

Definition *(Splitting Condition)*: The *splitting condition* holds for a *t-partition* in a graph G if the sum of the *ct-sizes* of the partitions and the join-size is strictly lesser than the *c-size* of G.

To obtain maximum computational benefits, we have to find a *t-partition* that minimizes the sum of the *ct-sizes* of the resulting partitions and the join-size. This general n-way split is NP-hard to find (easy to prove from the fact that finding the induced width is NP-hard). However, HCBFS employs a heuristic to decompose a large diagnosis problem into optimal sub-problems based on the topological structure of the system. It runs in polynomial time and is always assured of yielding computational benefits (albeit in sub-optimal amounts). The idea is to examine only a polynomial number of 2-way splits and choose the greediest one

ALGORITHM HCBFS (Graph $G = (V, E)$)
  T = *T-Graph* of G
  T' = Partition-Tree formed by m-ordering on moralized T
  E = Edges of T'
  GREEDYSPLIT (G, E)
END HCBFS

ALGORITHM GREEDYSPLIT (Graph G, Candidate-Splits B)
  b_k = BEST-SPLIT (G, B)
  IF (SPLITTING-CONDITION (G, b_k)) **THEN**
    (G_1, G_2) = PARTITION (G, b_k)
    $B_1 = \{b_i | b_i$ is on the same side of b_k as $G_1\}$
    $B_2 = \{b_i | b_i$ is on the same side of b_k as $G_2\}$
    GREEDYSPLIT (G_1, B_1)
    GREEDYSPLIT (G_2, B_2)
  END IF
END GREEDYSPLIT

Figure 3: Hierarchical Conflict-Directed Best First Search

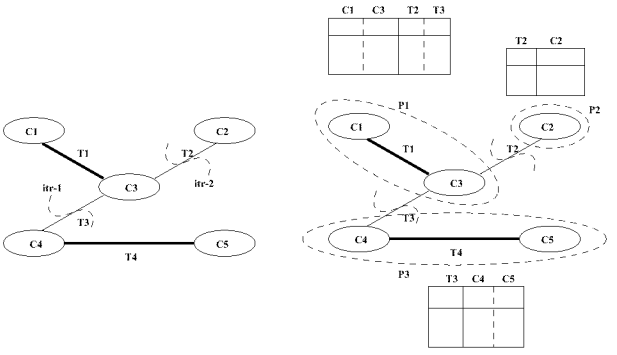

Figure 4: Illustrates the working of HCBFS to produce sub-problems. Thicker edges denote greater communication (*t-size*). P1, P2, P3 are the final partitions. The tables indicate the solutions to diagnosis sub-problems for all values of the surrounding communication variables.

if it satisfies the splitting condition. Such a splitting process is performed recursively until there is no more apparent scope for computational benefits. Interestingly enough, the candidate *t-partitions* that are examined are themselves derived using the m-ordering heuristics. Figure 3 presents the working of HCBFS; and Figure 4 illustrates its working on a small example. The following properties hold true for the HCBFS algorithm.

Property 1: The edges of T' maintain the *running intersection property* [Dechter, 1992] and hence the *t-nodes* constituting the communication variables on any edge form a valid *t-partition*.

Property 2: Let the *c-sizes* of the final partitions be $C_1 \cdots C_k$. The *c-size* of the original graph is therefore $\Pi_{i=1}^{i=k} C_i$. The first time we partition G, it must have been the case that (because of the splitting having to be satisfied) $\Pi_{i=1}^{i=k} C_i > S \times T + T \times R$ (T is the size of the communication; S and R are the *c-sizes* of the two resulting partitions with $S \times R = \Pi_{i=1}^{i=k} C_i$). In later iterations, the effective S and R are only made to decrease recursively and this essentially means that HCBFS is always *safe* in producing computational benefits.

Property 3: The total number of splits considered is clearly linear since they correspond to the edges of T'. Although there are two recursive calls to GREEDYSPLIT, the candidate set of edges that enter them are disjoint and hence GREEDYSPLIT is called only a linear number of times. This proves that the running time of HCBFS is polynomial.

Property 4: Choosing certain edges in a tree as splits results in a set of partitions that themselves form a tree with respect to the split edges (as illustrated in Figure 4). Since we know that optimization in a tree structured network is exponential in the *ct-size* of the largest partition, the complexity of diagnosis using HCBFS is exponential in this parameter.

4.1 Analysis and Implications of HCBFS

We briefly delve into the computational implications of HCBFS. HCBFS facilitates search in two ways. First, it reduces the effective search space by using the dynamic programming paradigm. Second, it propagates "easiness" in constraint checking. Constraint checking in general may not be computationally straightforward — it may often involve system "simulation" of some kind over an extended period of time. It can be noticed however, that constraint checking over the join space is a mere verification that two selected rows of the partition tables have similar values for their communication variables. By using HCBFS, the simulation-based constraint checks are "pushed" to smaller parts of the system (the partitions). Even for consistency checks that do not involve "simulation", implementing a TMS for each small partition is more effective (in terms of the complexity of data structures to be maintained) than one large TMS for the system as a whole.

HCBFS not only reduces the effective *un-amortized* search complexity for a diagnosis call, but also reduces the *amortized* complexity. The solutions to sub-problems occurring for diagnosis calls made in the past can be stored and used for future diagnosis calls when they need to solve the same sub-problems. Eventually, when all sub-problems for all values of communication variables have been solved at least once, a diagnosis call can be answered by doing a search only over the join-space of the partitions. This too (as argued before) is computationally easier than "simulation".

The dynamic programming idea of HCBFS can further be used to pre-process or compile the system description to facilitate diagnosis. Consider a partition of the graph representation of a physical setting. The idea is to solve the diagnosis problem for this partition for all values of the surrounding intermediate variables (*t-nodes*) and store the results. We can then treat this partition as a single physical component that can take any value (mode) corresponding to a combination of the values for each of its surrounding *t-nodes*. The associated probabilities would be derived from the results for the corresponding diagnosis sub-problems. This kind of precompilation of the system to treat partitions as components provides computational benefits only if their *t-size* is lesser than their *c-size* (which is often the case).

The space complexity associated with HCBFS has two components. One is the size of the tables associated with the sub-problems. This is referred to as the *table-space* complexity. It is easy to observe that the table space complexity is equal to the sum of the *t-sizes* over all partitions. Another component of the space requirement is the actual space required for the diagnosis algorithms to build the tables and compose them to answer a diagnosis call. This space requirement is identical to the running time complexities associated with solving and composing sub-problems. It is worth noting that the cost of implementing dynamic programming in HCBFS is reflected only in its table-space complexity.

HCBFS also leads to what are called hybrid approaches. These are techniques that combine conflict-based and coverage-based approaches [Kumar, to appear] to solve sub-problems and combine their solutions. Coverage-based algorithms are those that record conflicts and cast the diagnosis problem as a minimum weight hitting set problem [Kurien and Nayak, 2000]. Conflict-based approaches refer to the standard TMS-based algorithms like CBFS and QCBFS. In general, hybrid approaches do the following: (1) Employ the hierarchical partitioning algorithm to reduce the effective search space. (2) Employ one of coverage-based or conflict-based approaches for the sub-problems and the join space.

5 Comparison with Related Work

Related work on trying to leverage structure into the task of diagnosis can be found in [Darwiche, 1998], [Autio and Reiter, 1998], [Provan, 2001] etc. In [Darwiche, 1998], negation normal forms (NNF) are used to represent the consequence of $SD \cup OBS$. Subsequently, minimal cardinality diagnoses are extracted from them using a simple cost propagation and pruning algorithm. For such a procedure to be effective, it is important to ensure the decomposability of the NNF. Decomposability is achieved by partitioning SD to perform a case analysis on the shared atoms that do not appear among the observations. The partitioning choices are inspired by trying to produce a join-tree of the topological structure of the system much like the m-ordering heuristics. The complexity of the algorithm is exponential in the size of the hyper-nodes of

the join tree and linear in the number of such hyper-nodes.

There are at least three important ways in which this approach differs from ours. Firstly, this approach does not reason about probabilities but rather looks for minimal diagnoses (minimizes the number of faulty components). Secondly (and more importantly), it tries to produce diagnoses (minimal) by maintaining at each stage, a representation for all the consistent candidates. The optimization phase (of producing minimal candidates) occurs as a separate phase. Usually, we are not interested in all consistent diagnoses and trying to represent them at any stage when there could potentially be an exponentially large number of them can be a bottleneck. In our approach, the optimization and satisfaction phases are interleaved. This allows us to produce candidates as and when we want them, in decreasing order of their optimization values, and to prune the search space using both optimality- and satisfiability-reasoning. Thirdly, if the number of intermediate variables is too many, achieving decomposability in the NNF is exponential in the induced width of the moralized *T-Graph*; but since we are interested only in the behavior modes of components and not that of intermediate variables, the search space may be significantly reduced using our approach when the components are "tightly" coupled.

In [Provan, 2001] the idea of hierarchical diagnosis has a different meaning. It is based on the use of abstraction operators to define an abstraction hierarchy of the model (a lattice induced by a set of partitions of the system variables). A group of components and intermediate variables at a particular abstraction level are "merged" to form "abstract" components at a higher level with appropriately defined inputs, outputs and constraints relating them. A structural abstraction sc of subcomponents $c_1 \cdots c_k$ defines two modes of behavior for sc — $AB(sc)$ and $\neg AB(sc)$ with the constraint that $\neg AB(c_1) \cdots \neg AB(c_k) \rightarrow \neg AB(sc)$. Such an abstraction mechanism is useful only for isolating a group of components all of which cannot be behaving in their nominal modes (abstract models isolate diagnoses only at the abstract level, but more efficiently). At each level of abstraction we only define the nominal mode of behavior for the abstract component. The only other implicit mode is the faulty mode. This limits the scope of diagnosis even at the abstract levels. Under a combinatorial optimization formulation of the diagnosis problem, abstraction of $c_1 \cdots c_k$ to sc only defines what happens when all components $c_1 \cdots c_k$ are behaving in their most probable modes (nominal mode for sc). It does not say anything about what probabilities are associated or what happens with any of the other remaining exponentially large number of non-nominal modes. This makes diagnosis not only infeasible at more detailed levels, but also information-lossy at abstract levels.

6 Conclusions

In this paper, we employed the combinatorial optimization characterization of the diagnosis problem. We compared two different approaches that exploit different features of the problem: (1) naive m-ordering exploits the structure of the system by leveraging the causal dependencies among the variables (*T-Graph*) (2) CBFS exploits the fact that the output is uniquely determined for given inputs to a component behaving in a known mode, and that we are interested only in an assignment to the component modes of the system. We observed that naive m-ordering performs poorly when there is high interconnectedness among components and that CBFS performs poorly when there is low coupling. We proposed a computationally feasible algorithm called HCBFS (extending on CBFS) to achieve the best of both the worlds. HCBFS uses CBFS in tightly coupled parts of the system and m-ordering to identify them. We showed that HCBFS has many important implications on the complexity of diagnosis — reduces the un-amortized complexity of a diagnosis call, reduces amortized complexity of a diagnosis call by reusing computation done for sub-problems arising in past diagnosis calls, allows pre-compilation of the system description to facilitate diagnosis, and enhances hybrid algorithms. Finally, we compared and contrasted our work with somewhat related approaches.

References

[Autio and Reiter, 1998] Autio K. and Reiter R. Structural Abstractions in Model-Based Diagnosis. *In Proceedings of ECAI'98. Pages: 269-273.*

[Darwiche, 1997] Darwiche A. New Advances in Structure-Based Diagnosis: A Method for Compiling Devices. *In Proceedings of the Eighth International Workshop on Principles of Diagnosis (DX'1997). Pages: 35-42.*

[Darwiche, 1998] Darwiche A. Model-Based Diagnosis Using Structured System Descriptions. *Journal of Artificial Intelligence Research 8: 165-222.*

[Darwiche and Provan, 1997] Darwiche A. and Provan G. The Effect of Observations on the Complexity of Model-Based Diagnosis. *In Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI'97). Pages: 99-104.*

[Dechter, 1992] Dechter R. Constraint Networks. *Encyclopedia of Artificial Intelligence, second edition, Wiley and Sons, Pages: 276-285, 1992.*

[de Kleer *et al.*, 1992] de Kleer J., Mackworth A. K., and Reiter R. Characterizing Diagnoses and Systems. *Artificial Intelligence 56 (1992) 197-222.*

[de Kleer and Williams, 1989] de Kleer J. and Williams B. C. Diagnosis with Behavioral Modes. *In Proceedings of IJCAI'89. Pages: 104-109.*

[Forbus and de Kleer, 1992] Forbus K. D. and de Kleer J. Building Problem Solvers. *MIT Press, Cambridge, MA, 1992.*

[Hamscher *et al.*, 1992] Hamscher W., Console L., and de Kleer J. Readings in Model-Based Diagnosis. *Morgan Kaufmann, 1992.*

[Kumar, 2001] Kumar T. K. S. QCBFS: Leveraging Qualitative Knowledge in Simulation-Based Diagnosis. *Proceedings of the Fifteenth International Workshop on Qualitative Reasoning (QR'01).*

[Kumar, 2002] Kumar T. K. S. A Model Counting Characterization of Diagnoses. *Proceedings of the Thirteenth International Workshop on Principles of Diagnosis (DX'02).*

[Kurien and Nayak, 2000] Kurien J. and Nayak P. P. Back to the Future for Consistency-Based Trajectory Tracking. *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI'00).*

[Lucas, 1997] Lucas P. A Theory of Diagnosis as Hypothesis Refinement. *Proceedings of NAIC'97.*

[Nayak and Williams, 1997] Nayak P. P. and Williams B. C. Fast Context Switching in Real-time Propositional Reasoning. *In Proceedings of AAAI-97.*

[Provan, 2001] Provan G. Hierarchical Model-Based Diagnosis. *In Proceedings of the Twelfth International Workshop on Diagnosis (DX'01).*

[Struss, 1988] Struss P. Extensions to ATMS-based Diagnosis. *In J. S. Gero (ed.), Artificial Intelligence in Engineering: Diagnosis and Learning, Southampton, 1988.*

[Struss and Dressler, 1989] Struss P. and Dressler O. "Physical Negation" - Integrating Fault Models into the General Diagnosis Engine. *In Proceedings of IJCAI-89. Pages: 1318-1323.*

[Tarjan and Yannakakis, 1984] Tarjan R. E. and Yannakakis M. Simple Linear Time Algorithms to Test Chordality of Graphs, Test Acyclicity of Hypergraphs and Selectively Reduce Acyclic Hypergraphs. *SIAM J. Comput. 13, pages 566-576, 1984.*

[Williams and Nayak, 1996] Williams B. C. and Nayak P. P. A Model-Based Approach to Reactive Self-Configuring Systems. *In Proceedings of AAAI-96. Pages: 971-978.*